The JHU Machine Translation Systems for WMT 2016

The JHU Machine Translation Systems for WMT 2016

Shuoyang Ding, Kevin Duh, Huda Khayrallah, Philipp Koehn, and Matt Post
Center for Language and Speech Processing
Human Language Technology Center of Excellence
Department of Computer Science
Johns Hopkins University, Baltimore, MD
{dings,kevinduh,huda,phi,post}@jhu.edu

Abstract

This paper describes the submission of Johns Hopkins University for the shared translation task of ACL 2016 First Conference on Machine Translation (WMT 2016). We set up phrase-based, hierarchical phrase-based and syntax-based systems for all 12 language pairs of this year's evaluation campaign. Novel research directions we investigated include: neural probabilistic language models, bilingual neural network language models, morphological segmentation, and the attention-based neural machine translation model as reranking feature.

1 Introduction

The JHU 2016 WMT submission consists of phrase-based systems, hierarchical phrase-based systems, and syntax-based systems. In this paper we discuss features that we integrated into our system submissions. We also discuss the experiments we did with morphological pre-processing and neural reranking.

The JHU phrase-based translation systems for our participation in the WMT 2016 shared translation task[1] are based on the open source Moses toolkit (Koehn et al., 2007). We built upon strong baselines of the Edinburgh-JHU joint WMT submissions from the last year (Haddow et al., 2015), the Edinburgh syntax-based system submissions from the last year (Williams et al., 2015) as well as recent research in the field (Vaswani et al., 2013; Devlin et al., 2014). We also used the Apache Joshua translation toolkit (Post et al., 2015) to build hierarchical systems for two language tasks.

[1] http://www.statmt.org/wmt16

2 Moses Phrase-Based Systems

The phrase based system builds on the joint JHU-Edinburgh system from last year (Haddow et al., 2015). This year, we included Och clusters in various feature functions in the official submission. In addition, we included a large language model based on the CommonCrawl monolingual data and a neural network joint model.

2.1 Basic Configuration

We trained our systems with the following settings: a maximum sentence length of 80, grow-diag-final-and symmetrization of GIZA++ alignments, an interpolated Kneser-Ney smoothed 5-gram language model with KenLM (Heafield, 2011) used at runtime, hierarchical lexicalized reordering (Galley and Manning, 2008), a lexically-driven 5-gram operation sequence model (OSM) (Durrani et al., 2013) with 4 count-based supportive features, sparse domain indicator, phrase length, and count bin features (Blunsom and Osborne, 2008; Chiang et al., 2009), a distortion limit of 6, maximum phrase-length of 5, 100-best translation options, compact phrase table (Junczys-Dowmunt, 2012) minimum Bayes risk decoding (Kumar and Byrne, 2004), cube pruning (Huang and Chiang, 2007), with a stack-size of 1000 during tuning and 5000 during test and the no-reordering-over-punctuation heuristic (Koehn and Haddow, 2009). We optimize feature function weights with k-best MIRA (Cherry and Foster, 2012).

We used POS and morphological tags as additional factors in phrase translation models (Koehn and Hoang, 2007) for the German-English language pairs. We also trained target sequence models on the in-domain subset of the parallel corpus using Kneser-Ney smoothed 7-gram models. We used syntactic preordering (Collins et al., 2005)

Proceedings of the First Conference on Machine Translation, Volume 2: Shared Task Papers, pages 272–280,
Berlin, Germany, August 11-12, 2016.

Language Pair	Sentences
German–English	19,074
Czech–English	19,074
Finnish–English	1,500
Romanian–English	943
Russian-English	9,006
Turkish–English	500

Table 1: Tuning set sizes for phrase-based system

and compound splitting (Koehn and Knight, 2003) for the German-to-English systems. We did no language-specific processing for any other language.

The systems were tuned on a very large tuning set consisting of the test sets from 2008-2014, with a total of 19,074 sentences (see Table 1). We used news-test 2015 as development test set. Significantly less tuning data was available for Finnish, Romanian, and Turkish.

2.2 Och Clusters

As in last year's system, we use word classes in four feature functions: (i) the language model, (ii) the operation sequence model, (iii) the reordering model, and the (iv) sparse word translation features.

We generated Och clusters (Och, 1999) — a variant of Brown clusters — using `mkcls`. We have to choose a hyper parameter: the number of clusters. Our experiments and also prior work (Stewart et al., 2014) suggest that instead of committing to a single value, it is beneficial to use multiple numbers and use them in multiple feature functions concurrently. We used 50, 200, 600, and 2000 clusters, hence having 4 additional interpolated language models, 4 additional operation sequence models, 4 additional lexicalized reordering models, and 4 additional sets of sparse features.

The feature functions for word classes were trained exactly the same way as the corresponding feature functions for words. For instance, this means that the word class language model required training of individual models on the subcorpora, and then interpolation.

The computationally most expensive use of word clusters is in the language model, and to some degree the operation sequence model, both in terms of RAM and decoding speed. However, last year's experiments also showed that they are most effective there.

Language	Tokens	LM Size
Czech	6.7 billion	13GB
German	65.2 billion	107GB
English	65.1 billion	89GB
Finnish	2.9 billion	8GB
Romanian	8.1 billion	13GB
Russian	23.3 billion	41GB
Turkish	11.9 billion	23GB

Table 2: Sizes of the language model trained on the monomlingual corpora extracted from Common Crawl.

2.3 Huge Language Model

This year, large corpora of monolingual data were extracted from Common Crawl (Buck et al., 2014). We used this data to train 5-gram Kneser-Ney smoothed language models, pruning out 3–5 gram singletons. We trained these models with `lmplz`, as we did all other language models. We compressed the language models with KenLM with 4-bit quantization and use of the trie data structure.

The resulting size of the language model is listed in Table 2. The largest language model is the German model at 107GB, trained on 65.2 billion tokens, about an order of magnitude larger than previous data.

2.4 Neural Network Joint Model

The bilingual neural network language model, or neural network joint model for machine translation (NNJM), was first proposed in (Devlin et al., 2014). The basic idea is to construct neural language model as in (Vaswani et al., 2013), but include both the source and target side of the parallel corpus into the modeling context. Specifically, for a target word t_i within context $\mathcal{T}$ and $\mathcal{S}$, a (n + 2m + 1)-gram NNJM will model: $P(t_i \mid \mathcal{T}, \mathcal{S})$ where $\mathcal{T} = t_{i-n}, ..., t_{i-1}$, and $\mathcal{S} = s_{a_i-m}, ..., s_{a_i}, ..., s_{a_i+m}$ (a_i is the index of the word that is aligned to target word t_i).

We used the NPLM toolkit to build NNJMs for German-English, Romanian-English and Russian-English in both directions. We set the target side context window size to 5 and source side window size to 4. For all the NNJMs we built, the learning rate was set to 1.0 and we trained the models for 10 epochs. We kept all the other parameter values to their defaults.

2.5 Domain-Weighted Neural Network Probablistic Language Model

The neural probablistic language model (NPLM) was proposed by Bengio et al. (2003), but was not used inside the machine translation decoder until Vaswani et al. (2013) resolved the efficiency issues. It tries to approximate the same distribution as traditional language models with a feed forward neural network. Since discrete words are converted into continuous representations known as word embeddings, it has the potential to handle longer contexts without having to worry about issues with smoothing.

We used the NPLM toolkit[2] to build neural language models. Because of time and computation constraints we did not include these models in our final submission, but we experimented with different parameters for the relatively small Romanian monolingual data. We also tried different approaches to fine-tune the neural language model against the target side of English-Romanian tuning data, which will be discussed in this section.

The traditional way to handle domain relevance is to build language models for different domains of monolingual data separately, and then interpolate them by maximizing the probablity of a tuning set. This is because (1) the parallel data does not necessarily fit the domain of the monolingual data, and (2) querying several different language models would incur too much computation. But for NPLM, the non-linear layers make effective interpolation of different models less trivial.

To avoid interpolation and still accommodate domain adapation, we explored two solutions:

- consolidate all the monolingual data and train a large NPLM on the consolidated data
- fine-tune NPLM against the tuning corpus

The next natural question to ask is: how should the fine-tuning be done? We hereby propose three methods that we tried in our experiments:

1. Initialize with the weights obtained from training, go through the tuning corpus like training and back-propagate through all the weights in the network;
2. Like method 1, but only back-propagate through the last layer of the network. This could alleviate the problem of overfitting to the tuning data;
3. Take the interpolation weights $w_1, w_2, ..., w_n$ of the traditional language model trained on the same division of monolingual data with word count $c_1, c_2, ..., c_n$. Compute the normalized interpolation weights as follows:

$$\tilde{w_i} = \frac{w_i}{c_i}$$

In place of weighting and combining multiple language models, we will weight the training data and train a single language model on the weighted data. For example, if language model trained on corpus 1 has weight 1.0 and language model trained on corpus 2 has weight 1.5, we will repeat corpus 1 twice and corpus 2 three times. We then train the NPLM on this repeated and consolidated corpus.

Note that this method only used tuning data implicitly during the process of obtaining interpolation weights for the traditional language model.

System	newsdev2016b
baseline	23.1
w/o untuned nplm on all data	23.5 (+.4)
w/o untuned nplm on setimes2	23.2 (+.1)
w/o all data nplm + method 1	23.4 (+.3)
w/o all data nplm + method 2	23.8 (+.7)
w/o all data nplm + method 3	24.0 (+.9)

Table 3: Comparison of English-Romanian translation results of baseline system and systems with tuned/untuned NPLMs

Table 3 showed our English-Romanian translation results with NPLM trained on Romanian monolingual data. For method 3, we obtained the interpolation weights by first building language model on Europarl and setimes data using KenLM, and then interpolate the two language model against `newsdev2016a` data using SRILM. According to the interpolation weights obtained, we repeated setimes2 data for 108 times and did not repeat Europarl data before consolidation. Both the training and tuning were run for 5 epochs with a learning rate[3] of 0.25.

[2] `http://nlg.isi.edu/software/nplm/`

[3] The original NPLM paper used learning rate of 1. But in our experiments any learning rate more than 0.25 would cause `inf` values in the final parameters.

Language Pair	Best 2015	Baseline	w/ clusters	w/ CC LM	w/ both	w/ NNJM	w/ all & ttl100
English-Turkish	-	7.8	8.2 +0.3	9.4 +1.6	8.9 +1.1		
Turkish-English	-	14.0	14.3 +0.3	13.9 –0.1	14.1 +0.1		
English-Finnish	15.5	11.9	12.6 +0.7	12.2 +0.3	12.9 +1.0		
Finnish-English	19.7	16.5	16.9 +0.4	16.4 –0.1	16.9 +0.4		
English-Romanian	-	23.4	24.6 +1.2	23.4 +0.0	23.5 +0.1	23.7 +0.4	23.5 +0.1
Romanian-English	-	32.0	32.5 +0.5	32.5 +0.5	32.8 +0.8	32.0 +0.0	32.8 +0.8
English-Russian	24.3	23.9	25.0 +1.1	23.9 +0.0	24.9 +1.0	24.4 +0.5	25.2 +1.3
Russian-English	27.9	27.5	28.3 +0.7	28.1 +0.6	28.2 +0.7	27.8 +0.3	28.7 +1.2
English-Czech	18.8	18.2	19.2 +1.0	18.8 +0.6	19.6 +1.4		
Czech-English	26.2	27.0	27.7 +0.6	27.7 +0.7	28.1 +1.1		
English-German	24.9	22.7	23.0 +0.3	22.5 –0.2	22.7 +0.0	22.6 –0.1	22.9 +0.2
German-English	29.3	29.0	29.6 +0.6	29.6 +0.6	29.9 +0.9	29.6 +0.6	30.0 +1.0

Table 4: Phrase-Based Systems

NPLM generally improves the translation performance, but tuning method 1 does not help compared to the untuned version of the NPLM. Both method 2 and method 3 improve the performance even further. What's also interesting is that although we repeated setimes2 data so many times, solely building NPLM on setimes2 does not give a comparable performance, hence the coverage advantage as introduced by adding more datasets still makes a difference.

2.6 Results

Table 4 summarizes the impact of the contributions described in the preceeding sections. On their own, the use of Och clusters helped for all language pairs, the huge language model for almost all language pairs, and the neural network joint model for almost all language pairs. The gains are partially additive.

The biggest consistent gains are observed on Czech and Russian, in both directions, and for German–English — but not English–German. In the past we noted that the baseline English–German system, which includes a part-of-speech language model, is not helped much by the Och clusters. However, we are surprised by the lack of support from the huge language model.

For the other language pairs, the smaller tuning sets and hence higher variance in test scores make the results harder to interpret. The use of the huge language model gives mixed results, and gains are often not only not additive, but having more features hurts. See especially English–Romanian and Turkish–English where the best system does not include the huge language model, and Turkish–English where the best system only uses the huge language model.

Still, for all language pairs, the use of all features (and a translation table limit of 100) allowed us to outperform the strong baseline by +.1 to +1.3, for most language pairs around +1 BLEU.

3 Morphological Decomposition

We explored various methods for handling complex morphology. While we only apply these methods to Turkish, the methods are language independent. All of these methods were used and evaluated in the context of a Moses phrase-based system, as described in section 2.1.

We experimented with three segmentation algorithms: Morfessor (Virpioja et al., 2013), ChipMunk (Cotterell et al., 2015), and Byte-Pair encoding (Sennrich et al., 2015). .

Morfessor implements a set of segmentation algorithms designed for languages with concatenative morphology (where additional morphemes are added to convey meaning, but the stem and existing morphemes are not typically altered). Turkish falls in this category.

We focus on the Morfessor baseline algorithm, and use it without supervised word segmentations. While the segmentation may resemble linguistic segmentation, this is not guaranteed.

ChipMunk is an algorithm for segmenting words into morphemes and labeling those segments. It jointly models segmentation and labeling of the segments. While we do not use the label information, the labeling of segments is designed to reduce certain segmentation errors. For example, it prevents a prefix from directly attaching to a suffix, which prevents the segmentation of reed into re-ed. Since we choose not to rely on linguistic knowledge of Turkish, we use the pre-trained model with the tag level parameter set to 2.

Byte Pair Encoding (Gage, 1994) is a compression algorithm that recursively replaces frequent

Language	Threshold	Token count
Turkish	none	8806
Turkish	0	9606
Turkish	2	9935
Turkish	5	10169
Turkish	10	10416
Turkish	20	10720
English	none	11514

Table 5: Token counts for different thresholds for Morfessor segmentation

consecutive bytes with a symbol that does not occur elsewhere. Each such replacement is called a merge, and the number of merges is a tunable parameter. The original text can be recovered using a lookup-table. Sennrich et al. (2015) applied this to word segmentation, and demonstrate its success at solving the large vocabulary problem in neural machine translation.

To create our training data for the Morfessor and ChipMunk experiments, we augment the original training data with a second copy that has been segmented. For the tuning and test data, we only segment words that occur infrequently. This allows frequent words to be translated directly, but also allows the system to learn from the subword units of all words, including frequent ones.

The number and type of subword units in each word segmented by byte pair encoding is dependent on the number of merges performed. Since byte pair encoding segments words into the largest unit found in the table, and common words will occur in the table, this means that no subword information is extracted from common words. We set the number of merges to 50,000, and report the result, but did not explore it further. Perhaps a smaller number of merges would force the segmentation of more frequent words.

Table 5 shows the number of tokens using Morfessor and different segmentation strategies, as well as the number of tokens in the English parallel text. We show the number of tokens here because a common rationale for segmenting morphological rich languages is to balance the number of tokens.

BLEU scores for different amounts of segmentation are in Table 6. We report cased score on the development test set described in section 2.1. We see the best improvements with the ChipMunk segmentation and a rare word replacement threshold of 20. We also see gains with the fully unsupervised Morfessor segmentation.

Method	Processing	Thresh.	BLEU
baseline	-	-	13.9
Byte-Pair	preprocessing	-	13.7
Chipmunk	replace-rare	2	14.3
Chipmunk	replace-rare	10	14.9
Chipmunk	replace-rare	20	**15.4**
Chipmunk	replace-rare	20	14.7
Morfessor	replace-rare	0	13.5
Morfessor	replace-rare	2	13.7
Morfessor	replace-rare	5	14.0
Morfessor	replace-rare	10	**14.1**
Morfessor	replace-rare	20	14.2

Table 6: Turkish - English morphology results on newsdev2016b

None of these results completed in time for our official submission, which used unsegmented text.

4 Neural Sequence Model Reranking

We also experimented with N-best reranking using a neural sequence model. The motivation is to exploit the efficiency of standard phrase-based models for generating N-best hypotheses, combined with the modeling power of neural methods for scoring. This hybrid approach demonstrated positive results in, e.g. (Cohn et al., 2016).

First, we train neural attention models (Bahdanau et al., 2015) on our training bitext in both forward and reverse directions. For example, in the Russian-English task we would obtain a forward model $p(en|ru)$ that scores English hypotheses given Russian input, and a reverse model $p(ru|en)$ that scores the Russian input given an English hypothesis. We used the TensorFlow implementation (Abadi et al., 2015), with 2-layers of LSTMs and 1024 hidden units each; other hyperparameters use default settings. The training bitext is preprocessed so that top 10k vocabulary in terms of frequency are kept as is, frequency 1 singletons are mapped to an unknown word token (UNK), and all remaining mid-frequency words are mapped to one of 600 Brown clusters. Note that our vocabulary list of 10,601 is considerably less than those used in existing neural MT literature (e.g. 30k in (Bahdanau et al., 2015)). This is a trade-off between modeling power vs. training time, and we felt that a smaller vocabulary may

be sufficient since our model only scores existing translations and do not need to generate.

Second, we generate distinct 50-best lists on our tuning set using a baseline Moses phrase-based system in Table 4. Each hypothesis is scored by the forward and reverse neural models, and together with the original Moses decoder score and word penalty features (for a total of 4 features), we reran MERT training and obtained optimal linear weights. This is the reranker system used for reranking test N-best lists.

Preliminary results with this approach were inconclusive. For example, on the Russian-English newstest2015, the BLEU score is 27.27 for 1-best vs. 27.31 for reranking. On German-English newstest2015, the BLEU score is 28.12 for 1-best and 28.22 for reranking. (Note, these results did not include a post-processing step and are thus not comparable to the numbers in Table 4). Analysis of the output showed that our rerankers appear conservative and do not frequently chooses hypotheses different from the 1-best. We believe more experimentation with different neural model hyperparameters is necessary. Future work also includes experimentation on larger N-best lists and comparison with direct 1-pass translation using neural models. This reranker was not included in the final submission.

5 Moses Syntax-based Systems

In this section we discuss our setup for the three string-to-tree syntax-based systems we submitted for German-English, English-German, and Turkish-English. Unless metioned in this section, we use the same setup as the baseline Moses phrase-based systems described in Section 2.

5.1 Preprocessing

Since we are building string-to-tree syntax-based models, we need to parse the target side of the parallel corpus before training. For our syntax-based models we used the Berkeley parser to parse the English side of the German-English and Turkish-English parallel corpus, and ParZu (Sennrich et al., 2009) to parse the German side of the English-German parallel corpus. For the Berkeley parser we used the default English grammar `eng_sm6.gr`. For ParZu we used clevertagger[4] as the POS-tagger, SMOR[5] as the morphological analyzer, and used `zmorge-20140521-smor_newlemma.ca` as the model for morphological analysis. We pass the all corpus for syntax-based models through `deescape-special-chars.perl` before parsing to avoid formatting problems.

In addition to parsing, for English-German syntax-based model, we also used `hybrid_compound_splitter.py` to split the compound words in German, as in Edinburgh's WMT 2015 submission (Williams et al., 2015)[6]. We used the same morphological analyzer model as used for ParZu.

5.2 Feature Scores

The most significant difference between the syntax-based model and the phrase-based model is that the translation model score is calculated by SCFG translation rule scores instead of phrase pair scores. Specifically, a SCFG translation rule r is denoted as:

$$\mathcal{L} \rightarrow < \mathcal{S}, \mathcal{T}, \mathcal{A} >$$

where $\mathcal{L}$ is the left-hand side label as shared by both sides of the translation, $\mathcal{S}$ and $\mathcal{T}$ is a sequence of terminal and non-terminal nodes on the source and target side, respectively. $\mathcal{A}$ denotes the alignment between the source and target nodes. Given a derivation $\mathcal{D}$ that generates the sentence pair, the forward and inverse translation model score is:

$$fwd = \prod_{\mathcal{L} \rightarrow < \mathcal{S}, \mathcal{T}, \mathcal{A} > \in \mathcal{D}} P(\mathcal{L}, \mathcal{T} \mid \mathcal{S}, \mathcal{A})$$

$$inv = \prod_{\mathcal{L} \rightarrow < \mathcal{S}, \mathcal{T}, \mathcal{A} > \in \mathcal{D}} P(\mathcal{S} \mid \mathcal{L}, \mathcal{T}, \mathcal{A})$$

Apart from these two scores, we also added unknown word soft matching features and glue rule penalties. We also kept the lexical translation scores, word penalties and phrase penalties etc. as in the phrase-based translation models.

5.3 Configurations

To avoid problems during syntax-based rule extraction and decoding, we removed all the factors such as lemma and POS-tags and only use word during the training phase.

We used the GHKM rule extractor implemented in Moses to extract SCFG rules from the parallel

[4] `https://github.com/rsennrich/clevertagger`

[5] `http://kitt.ifi.uzh.ch/kitt/zmorge/`

[6] `https://github.com/rsennrich/wmt2014-scripts`

corpus. We set the maximum number of nodes (except target words) in the rules (`MaxNodes`) to 20, maximum rule depth (`MaxRuleDepth`) to 5, and the number of non-part-of-speech, non-leaf constituent labels (`MaxRuleSize`) to 5, and we allowed unary rules to appear in the extracted phrases. We also limited the maximum number of lexical items in a rule to 5.

To avoid excessive use of glue rules, we fixed the feature weight for glue rules as -99 during tuning step.

6 Joshua Systems

We also used the Apache Joshua translation toolkit[7] to build Hiero systems for two languages: English–Finnish and English–Turkish. The default settings were used for the Thrax grammar extractor: Hiero rules were extracted from spans as large as 10 words, and applied at decoding time to spans as long as 20 words. We used all of the provided bitext for both language tasks. All systems used three language models: one built on the target side of the bitext, another built on all available common-crawl monolingual data, and a third, class-based 9-gram language model built on the target side of the bitext after applying Brown clustering (k=2,000). Each of these received a separate weight. We tuned with k-best batch MIRA (Cherry and Foster, 2012).

Case is important for the human evaluation, and its proper handling has received some attention. Instead of applying a method such as truecasing (Lita et al., 2003), we use the following heuristic for these languages. First, we convert all data to lowercase at training time, so the models are learned in lowercase. At test time, input words are lowercased and marked with a tag denoting whether each source word was (a) lowercase, (b) Capitalized, or (c) ALL UPPERCASE. These case markings are then projected to the target words through the word-level alignments stored with grammar rules. This worked well for the language pairs under consideration, though it would obviously not work for all language pairs.

We also employed one small trick with Turkish punctuation: after removing whitespace inside balanced single-quotes, we remove the space from both sides of remaining single-quotes. This captured a common pattern and resulted in a small BLEU score gain that helped propel the system into first place (by cased BLEU). Although the English–Turkish system had the highest BLEU score, however, it was in the fifth cluster in the manual evaluation. The English–Finish system did not perform well by either metric.

[7] `http://joshua.incubator.apache.org/`

Language Pair	Phrase	Syntax	Joshua
English-Turkish	9.2	-	**9.8**
Turkish-English	12.9	**13.9**	-
English-Finnish	**13.8**	-	11.9
Finnish-English	19.1	-	-
English-Romanian	23.5	-	-
Romanian-English	32.2	-	-
English-Russian	24.0	-	-
Russian-English	27.9	-	-
English-Czech	23.6	-	-
Czech-English	30.4	-	-
English-German	**28.3**	27.3	-
German-English	**34.5**	32.3	-

Table 7: Official scores of all submission on newstest2016 (cased BLEU).

7 Conclusion

Our submissions are summarized in Table 7. We submitted phrase-based systems for all 12 language pairs, syntax-based systems for 3 and Joshua hierarchical systems for 2 language pairs. For the low resource Turkish–English language pairs, the latter systems outperformed the phrase-based submission.

Compared to submissions from other groups, our performance is solid. In terms of neural machine translation components, we have seen gains from the use of the NNJM (Devlin et al., 2014) as a feature function but not in re-ranking with a sequence to sequence model. Given the success of these components in other systems, we will target their use in the future.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden,

Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, (3):1137–1155.

Phil Blunsom and Miles Osborne. 2008. Probabilistic inference for machine translation. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 215–223, Honolulu, Hawaii, October. Association for Computational Linguistics.

Christian Buck, Kenneth Heafield, and Bas Van Ooyen. 2014. N-gram counts and language models from the common crawl. *LREC*, 2:4.

Colin Cherry and George Foster. 2012. Batch tuning strategies for statistical machine translation. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 427–436, Montréal, Canada, June. Association for Computational Linguistics.

David Chiang, Kevin Knight, and Wei Wang. 2009. 11,001 New Features for Statistical Machine Translation. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 218–226, Boulder, Colorado, June. Association for Computational Linguistics.

Trevor Cohn, Cong Duy Vu Hoang, Ekaterina Vymolova, Kaisheng Yao, Chris Dyer, and Gholamreza Haffari. 2016. Incorporating structural alignment biases into an attentional neural translation model. In *Proceedings of NAACL-16*.

Michael Collins, Philipp Koehn, and Ivona Kucerova. 2005. Clause restructuring for statistical machine translation. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 531–540, Ann Arbor, Michigan, June. Association for Computational Linguistics.

Ryan Cotterell, Thomas Müller, Alexander Fraser, and Hinrich Schütze. 2015. Labeled morphological segmentation with semi-markov models. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning*, pages 164–174, Beijing, China, July. Association for Computational Linguistics.

Jacob Devlin, Rabih Zbib, Zhongqiang Huang, Thomas Lamar, Richard M Schwartz, and John Makhoul. 2014. Fast and Robust Neural Network Joint Models for Statistical Machine Translation. In *ACL*, pages 1370–1380.

Nadir Durrani, Alexander Fraser, Helmut Schmid, Hieu Hoang, and Philipp Koehn. 2013. Can markov models over minimal translation units help phrase-based SMT? In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, Sofia, Bulgaria, August. Association for Computational Linguistics.

Philip Gage. 1994. A new algorithm for data compression. *C Users J.*, 12(2):23–38, February.

Michel Galley and Christopher D. Manning. 2008. A simple and effective hierarchical phrase reordering model. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 848–856, Honolulu, Hawaii, October.

B Haddow, M Huck, A Birch, and N Bogoychev. 2015. The Edinburgh/JHU Phrase-based Machine Translation Systems for WMT 2015. In *WMT*.

Kenneth Heafield. 2011. Kenlm: Faster and smaller language model queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 187–197, Edinburgh, Scotland, United Kingdom, July.

Liang Huang and David Chiang. 2007. Forest rescoring: Faster decoding with integrated language models. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 144–151, Prague, Czech Republic, June. Association for Computational Linguistics.

M. Junczys-Dowmunt. 2012. A phrase table without phrases: Rank encoding for better phrase table compression. In Mauro Cettolo, Marcello Federico, Lucia Specia, and Andy Way, editors, *Proceedings of th 16th International Conference of the European Association for Machine Translation (EAMT)*, pages 245–252.

Philipp Koehn and Barry Haddow. 2009. Edinburgh's Submission to all Tracks of the WMT 2009 Shared Task with Reordering and Speed Improvements to Moses. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, pages 160–164, Athens, Greece, March. Association for Computational Linguistics.

Philipp Koehn and Hieu Hoang. 2007. Factored translation models. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 868–876, Prague, Czech Republic, June. Association for Computational Linguistics.

Philipp Koehn and Kevin Knight. 2003. Empirical methods for compound splitting. In *Proceedings of Meeting of the European Chapter of the Association of Computational Linguistics (EACL)*.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: open source toolkit for statistical machine translation. In *ACL*. Association for Computational Linguistics.

Shankar Kumar and William J. Byrne. 2004. Minimum bayes-risk decoding for statistical machine translation. In *HLT-NAACL*, pages 169–176.

Lucian Vlad Lita, Abe Ittycheriah, Salim Roukos, and Nanda Kambhatla. 2003. Truecasing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 152–159. Association for Computational Linguistics.

Franz Josef Och. 1999. An Efficient Method for Determining Bilingual Word Classes. In *Proceedings of the 9th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 71–76.

Matt Post, Yuan Cao, and Gaurav Kumar. 2015. Joshua 6: A phrase-based and hierarchical statistical machine translation system. *The Prague Bulletin of Mathematical Linguistics*.

Rico Sennrich, Gerold Schneider, Martin Volk, and Martin Warin. 2009. A new hybrid dependency parser for german. *Proceedings of the German Society for Computational Linguistics and Language Technology*, pages 115–124.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural machine translation of rare words with subword units. *CoRR*, abs/1508.07909.

Darlene Stewart, Roland Kuhn, Eric Joanis, and George Foster. 2014. Coarse split and lump bilingual language models for richer source information in SMT. In *Proceedings of the Eleventh Conference of the Association for Machine Translation in the Americas (AMTA)*, volume 1, pages 28–41.

Ashish Vaswani, Yinggong Zhao, Victoria Fossum, and David Chiang. 2013. Decoding with Large-Scale Neural Language Models Improves Translation. In *EMNLP*, pages 1387–1392. EMNLP, October.

Sami Virpioja, Peter Smit, Stig-Arne Grönroos, Mikko Kurimo, et al. 2013. Morfessor 2.0: Python implementation and extensions for morfessor baseline.

Philip Williams, Rico Sennrich, Maria Nadejde, Matthias Huck, and Philipp Koehn. 2015. Edinburgh's Syntax-Based Systems at WMT 2015. In *WMT*, September.